GenRad
futuredata

# ADVANCED MICROPROCESSOR DEVELOPMENT SYSTEM EMULATOR REFERENCE MANUAL

2300-5002-00

MICROCOMPUTER SYSTEMS

# ADVANCED MICROPROCESSOR DEVELOPMENT SYSTEM EMULATOR REFERENCE MANUAL

2300-5002-00

GenRad/Development Systems Division
5730 Buckingham Parkway
Culver City, CA 90230

© GenRad/Development Systems Division 1981

TABLE OF CONTENTS

1 Introduction

2 Emulator Features

3 Prior to Using the ICE

4 Step by Step Debugging

5 Emulator Commands

6 Debugger Commands

7 Installation

The In-Circuit Emulator (ICE) is a hardware feature that enables the microprocessor in the AMDS to emulate the microprocessor in the user system. A connector on the end of the ICE cable is plugged directly into the socket on the user prototype for the microprocessor. The purpose of an ICE is threefold. First the ICE can used to substitute RAM for the PROM/ROM in the user system. The RAM can be much more easily modified than the ROM/PROM thus facilitating changes in user programs during the development phase. ICE also provides debugging facilities in the user enviornment. Features such as single-step, trace, breakpoints are all included in the ICE package. Lastly, the ICE package allows the user to gradually bring up the target system. The clocks, DMA facilities, interrupts and I/O and memory mapping can all be separately enabled.

When combined with the Universal Logic Analyzer, the ICE -- analyzer combination provides 3 complex breakpoint registers with up to 24 address, 16 data and 8 control lines. The analyzer also allows real time tracing of 256 bus transactions.

The ICE package consists of the following elements

(1) An In-Circuit Emulator interface card,
(2) Cable(s) to connect the interface card to the processor card,
(3) An In-Circuit Emulator Probe assembly.

Initially the AMDS is used to debug and develop the user software. Then the ICE is used to verify the proper operation of the user hardware. The designer may switch clocks, control lines, memory from the AMDS to the user (target) system. As various elements of the user system are verified, more and more of these elements are utilized until, finally, the ICE connector is removed. This, step by step, procedure is extremely effective, since only one element, at a time, is being tested.

All of the AMDS debugger commands are available to the user during emulator operation. Thus the user is able to examine and alter memory and registers, start a program, set breakpoints, single-step, trace, and load and dump object programs.

The 8080 ICE has the following salient features:

(1) Real-time emulation and debugging at 2Mhz clock cycles using high-speed static RAM.

(2) Separate control over clock source, DMA, Interrupt and I/O, and refresh enable.

(3) Memory mapping in 8K blocks by jumper removal.

(4) Hardware breakpoint, single step, trace, and 4 software breakpoints.

(5) Logic analyzer adds:
- 3 complex breakpoints
  16 address, 8 data, Memory/IO, R/W, Instruction fetch, DMA cycle and 4 external conditions.
- trace qualifier
- 256 deep trace memory.

(6) The hardware breakpoint, software breakpoint, logic analyzer breakpoint, single-step, and trace all use the interrupt vector at X'30' (RESTART 6). The BREAK key uses the interrupt vector at X'20' (RESTART 4).

(7) A system with 48 kilobytes of RAM provides 28 kilobytes of ROM/PROM simulator memory from X'0000' to X'6FFF'.

(8) The user memory space is from X'0000' to X'5FFF'.

(9) All I/O ports are available to the user with the following exceptions:

X'1F', X'3F', X'5F', X'7F', X'9F', X'BF', X'C0-CE', X'D8-D9', X'DF', X'F0-F6', and X'F8-FF'.

The upgraded 8080-1 ICE has the following salient features:

(1) Real-time emulation and debugging at 2Mhz clock cycles using high-speed static RAM.

(2) Separate control over clock source, DMA, Interrupt and I/O, memory map and refresh enable.

(3) Memory mapping in 256 byte blocks under software control.

(4) Hardware breakpoint, single step, trace, and 4 software breakpoints.

(5) Logic analyzer adds:
- 3 complex breakpoints
  16 address, 8 data, Memory/IO, R/W, Instruction fetch, DMA cycle and 4 external conditions.
- trace qualifier
- 256 deep trace memory.

(6) The hardware breakpoint, software breakpoint, logic analyzer breakpoint, single-step, BREAK key and trace all use the interrupt vector at X'30' (RESTART 6).

(7) A system with 48 kilobytes of RAM provides 28 kilobytes of ROM/PROM simulator memory from X'0000' to X'6FFF'. A system with 64 kilobytes of RAM provides 44 kilobytes of ROM/PROM simulator memory X'0000' to X'8FFF' and X'E000' to X'FFFF'.

(8) The user memory space is from X'0000' to X'FFFF', with 256 bytes for the resident debugger communication block at X'D400' to X'D4FF'.

(9) All I/O ports are available to the user except output port X'F6'.

The 8085 ICE has the following salient features:

(1) Real-time emulation and debugging at 10Mhz clock cycles (5Mhz bus cycles) using high-speed static RAM.

(2) Separate control over clock source, DMA, Interrupt and I/O, memory map and refresh enable.

(3) Memory mapping in 256 byte blocks under software control.

(4) Hardware breakpoint, single step, trace, and 4 software breakpoints.

(5) Logic analyzer adds:
- 3 complex breakpoints
  16 address, 8 data, Memory/IO, R/W, Instruction fetch, DMA cycle and 4 external conditions.
- trace qualifier
- 256 deep trace memory.

(6) The hardware breakpoint, BREAK key , logic analyzer breakpoint, single-step, and trace all use the interrupt vector at X'2C' (RESTART 5.5). The software breakpoints use the interrupt vector at X'30' (RESTART 6).

(7) A system with 48 kilobytes of RAM provides 28 kilobytes of ROM/PROM simulator memory from X'0000' to X'6FFF'. A system with 64 kilobytes of RAM provides 44 kilobytes of ROM/PROM simulator memory from X'0000' to X'8FFF' and X'E000' to X'FFFF'.

(8) The user memory space is from X'0000' to X'FFFF', except for a 256 byte resident debugger communication block from X'D400' to X'D4FF'.

(9) All I/O ports are available to the user except output port X'F6'.

The Z80 ICE has the following salient features:

(1) Real-time emulation and debugging at 4Mhz clock cycles using high-speed static RAM.

(2) Separate control over clock source, DMA, Interrupt and I/O, memory map and refresh enable.

(3) Memory mapping in 256 byte blocks under software control.

(4) Hardware breakpoint, single step, trace, and 4 software breakpoints.

(5) Logic analyzer adds:
- 3 complex breakpoints
  16 address, 8 data, Memory/IO, R/W, Instruction fetch, DMA cycle and 4 external conditions.
- trace qualifier
- 256 deep trace memory.

(6) The hardware breakpoint, BREAK key, logic analyzer, breakpoint, single-step, and trace all use the interrupt vector at X'66' (Non-Maskable Interrupt). The software breakpoint uses the interrupt vector at X'30' (RESTART 6).

(7) A system with 48 kilobytes of RAM provides 28 kilobytes of ROM/PROM simulator memory from X'0000' to X'6FFF'. A system with 64 kilobytes of RAM provides 44 kilobytes of ROM/PROM simulator memory from X'0000' to X'8FFF' and X'E000' to X'FFFF'.

(8) The user memory space is from X'0000' to X'FFFF', except for a resident 256 byte debugger communication block at X'D400'to X'D4FF'.

(9) All I/O ports are available to the user except for output port X'F6'.

The 6800 ICE has the following salient features:

(1) Real-time emulation and debugging at 1Mhz clock cycles using high-speed static RAM.

(2) Separate control over clock source, DMA, Interrupt and I/O, memory map and refresh enable.

(3) Memory mapping in 256 byte blocks under software control.

(4) Hardware breakpoint, single step, trace, and 4 software breakpoints.

(5) Logic analyzer adds:
- 3 complex breakpoints
  16 address, 8 data, Memory/IO, R/W, Instruction fetch, DMA cycle and 4 external conditions.
- trace qualifier
- 256 deep trace memory.

(6) The hardware breakpoint, BREAK key, logic analyzer, breakpoint, single-step, and trace all use the interrupt vector at X'FFFC' (NMI). The software breakpoint uses the vector at X'FFFA' (SWI).

(7) A system with 48 kilobytes of RAM provides 24 kilobytes of ROM/PROM simulator memory from X'0000' to X'5FFF'. A system with 64 kilobytes of RAM provides 40 kilobytes of ROM/PROM simulator memory from X'0000' to X'7FFF', and X'C000' to X'DFFF'.

(8) The user memory space is from X'0000' to X'FFFF' with the exception of a resident debugger communication block at X'____' to X'____' and emulator control ports at X'FFF4' to X'FFF7'.

The 6802 ICE has the following salient features:

(1) Real-time emulation and debugging at 2Mhz clock cycles using high-speed static RAM.

(2) Separate control over clock source, HALT, Interrupt and I/O, memory map and refresh enable.

(3) Memory mapping in 256 byte blocks under software control.

(4) Hardware breakpoint, single step, trace, and 4 software breakpoints.

(5) Logic analyzer adds:
- 3 complex breakpoints
  16 address, 8 data, Memory/IO, R/W, Instruction fetch, DMA cycle and 4 external conditions.
- trace qualifier
- 256 deep trace memory.

(6) The hardware breakpoint, BREAK key, logic analyzer breakpoint, single-step, and trace all use the interrupt vector at X'FFFC' (NMI). The software breakpoint uses the interrupt vector at X'FFFA' (SWI).

(7) A system with 48 kilobytes of RAM provides 24 kilobytes of ROM/PROM simulator memory from X'0000' to X'5FFF'. A system with 64 kilobytes of RAM provides 40 kilobytes of ROM/PROM simulator memory from X'0000' to X'7FFF' and X'C000' to X'DFFF'.

(8) The user memory space is from X'0000' to X'FFFF' except for a 256 byte resident debugger communication block from X'____' to X'____' and emulator control ports from X'FFF4' to X'FFF7'.

In order to provide an emulator that is a close approximation of the chip it is necessary to provide minimum isolation between the the user system and the ICE logic. Thus there are no terminations on the data lines, address lines, control lines and clock lines at the ICE probe connections. This means that the ICE will be sensitive to noise on the data and control lines of the user system, just as the microprocessor chip would be. Many of the problems associated with microprocessor based systems are associated with noisy data and control lines. In a system where TTL bus drivers are used to buffer the data bus, noise can be generated in the process where the data bus is turned around to switch from read to write. Schottky TTL chips, such as, 8T97, 8T98, 8T26, 8216, 8226, 74LS240, 74LS244 and 74LS245 are capable of generating very narrow switching tranisients (less than 5 nanoseconds). These can, of course, cause a significant amount of crosstalk between data and control lines when the data lines switch. In addition, control signals used to "turn around" bi-directional data buses may generate bus conflicts at the instants the directions are changed. This can also cause significant cross talk.

One further caution concerns the connection of the grounds between the AMDS and ICE on one hand and the user system on the other. The ICE logic ground is connected to the AMDS frame and to the third wire on the power cord at a single point in the AMDS on the system power supply. If the user system also has a third wire connection there will be a ground loop through the power line grounds that may generate considerable ground noise. Obviously this condition is to be avoided by disconnecting the user logic ground from the user chassis during the ICE debugging phase.

The user also needs to be aware of the location of the stack. The debugger will set the stack pointer to an area in the AMDS memory space that is read/write memory. This stack will be used both by the user software and by the debugger (to save and restore the registers at exit from and entry to the user program). If this memory location is mapped into the user space, then it must still be read/write memory or else the user program must load the stack pointer with a suitable address. The debugger will not operate correctly without a stack in read/write memory.

In order to develop hardware and software expeditiously, it is very desireable that the designer follow a step by step approach in validating the hardware and software at each step in the design process. The designer should keep in mind the basic goal of separating hardware and software problems. Thus, the designer will attempt to verify as much as possible of the hardware system by means of simple tests using unambiguous debugging programs. If this goal is kept in mind, then it should be possible to verify nearly all of the hardware system prior to executing the actual system program.

There are seven distinct steps in the design verification of a microprocessor based system. These are:

(1) Verify operation of the AMDS using clocks generated by the user system.
(2) Verify operation of the microprocessor when the user control lines are active (DMA request, Interrupt request(s), Wait, Reset, Ready, etc.).
(3) Verify basic operation of the system using very simple programs.
(4) Verification of user I/O and memory using diagnostic programs resident in the AMDS ROM/PROM simulator memory.
(5) Execution of the user program with all of the memory space in the AMDS and all of the I/O space in the user system.
(6) Execution of the user program with all of the memory space and all of the I/O space mapped into the user system.
(7) Execution of the user program with all memory, I/O and the microprocessor installed in the user system.

## Clock Test

The user should first verify that the clock timing, rise and fall times, and levels meet the requirements of the microprocessor manufacturer. They should be measured using a good oscilloscope and compared to the specification of the chip manufacturer. Then it should be possible to plug in the emulator probe and switch the clock line input(s) to the user system. Note, it will be immediately apparent if the clocks are not suitable since operation of the AMDS will halt.

If the system will not run using the user clocks, check (1) clock signals with the ICE probe plugged in, (2) possibility of ground loops.

## Operation of the System With User Inputs on the Control Lines

In general, microprocessor input control lines fall into three catagories: (1) Interrupt and Reset inputs, (2) DMA (Direct Memory Access) requests, and (3) asynchronous wait inputs. The ICE debuggers allow the user to separate control of the DMA requests from the Interrupt, Reset and Wait inputs, since debugging of DMA operation is so different from Interrupt and Reset functions. The user will have to refer to the command parameters as described in the M (Mode) command to determine the proper command to verify operation with the control inputs. Some of the considerations that user should be aware of, when he is verifying the control inputs are:

(1) The use of the Reset input by the debugger to clear the microprocessor registers. This is modifiable by the user to cause his reset sequence, however the user must remember to modify the reset vector or instruction to cause his sequence.

(2) The use of one or more of the microprocessor interrupt vectors, pins etc. by the debugger for hardware breakpoint, software breakpoint, trace, single step and break key. Again the user may modify this action, but he must be aware of the debugging features that may be lost.

(3) The user system cannot have an interrupt pending at the onset of user program execution, or that interrupt will be acknowledged immediately before any code is

executed, including any code to mask interrupts, set up interrupt vectors, set up interrupt prioritys etc. The "N" (NOINTERRUPT) command is very useful in this instance since it allows the the user to begin execution of his program with interrupts disabled. Then the initialization routines may be executed prior to unmasking the interrupt system. Of course, any use of Non-maskable interrupts cannot be blocked by using the "N" command.

(4) The AMDS uses DMA requests to refresh the CRT display and dynamic RAM. This will show up as gaps in the operation of the microprocessor while these requests are serviced. If the user is concerned about absolute system timing, it will be necessary to use static RAM and the "R" parameter with the mode command to inhibit DMA requests during user program execution.

## Verify Basic Operation of the Microprocessor in the User System

The user should now execute a simple program to verify opertion of the ICE in his system. Of course, the clocks and control lines should be switched to get their inputs from the user system. Very simple programs that can be easily analyzed for proper operation are: (1) one instruction programs that jump back to themselves, (2) simple I/O exercisers that read or write to a single port continously (these are two instruction programs that read or write to an I/O port and then jump back to themselves). (3) simple memory exercisers that read or write a single memory location and allow the user to scope a "chip select" line. The use of an I/O or Memory write is particularly convenient, since the microprocessor write strobe may be used as a scope sync to verify other signal lines.

If the user is unable to execute simple programs of this form, he should check: (1) for noisy input control lines (see section "Things to be Aware of Prior to Using an In-Circuit Emulator"), (2) for ground loops (see section "Things to be Aware of Prior to Using an In-Circuit Emulator"), (3) for proper clock signals (see section "Clock Test"), (4) for proper operation of the input control lines (see section "Operation of the System With User Inputs on the Control Lines).

## Diagnostic Program Execution

It is recommended that at this point the designer run very simple diagnostic programs. These programs may be as simple as sending repetitive input/output commands using a two instruction loop, where the first instruction is the I/O command and the second instruction is a jump to the first instruction. This will allow the designer to probe various points in the system and verify the occurance of decoded I/O commands. As a next step the programs can be altered so that all combinations of data are sent to output ports. The programs can be further modified to read an input port and write the results to an output port. These programs should be done one at a time for single input and output ports so that the results may be may be easily verified by interrupting the process and examining the AMDS register display.

The extent of these diagnostics will, of course, depend on the complexity of the user I/O system. Simple parallel input and output registers can be easily tested by loading bit patterns in to the accumulator and transferring them into the output registers and by reading various bit patterns form the input registers. Obviously, as more complex I/O functions are done, more complex diagnostic programs will have to be written. Thus it would not be uncommon for a complicated peripheral device, such as a floppy disk , to require ten to twenty pages of diagnostice program.

It is a good idea to verify the correct operation of RAM on the user prototype at this point. Simple programs that write the low order 8 address bits into the memory can be used to test the data lines and low order address lines. Of course, more sophisticated memory diagnostics are needed for full verification of the user memory.

## User Program Execution Using ROM/PROM Simulator Memory

The designer is now able to finally try to execute the system program, or at least parts of it, in ROM/PROM simulator memory. The system should behave as though it were operating in its final form.

There are some precautions that need to be observed. These are:

(1) Leave the debugging vectors alone, as long as possible. The vector locations are listed in the sections on

"Emulator Features" for the various processors.

(2) Be aware of the use of the memory space by the debugger. Again, the space available to the user is shown in the sections on "Emulator Features" for the various processors. Note: that the user can have his own memory in the same address space as the debugger. This memory must be installed in the user system and cannot be ROM/PROM simulator memory.

(3) The relocatble assembler and linker can be used to put the user program at any arbitrary location. This is convenient while the program is in ROM/PROM simulator memory.

## User Program Execution Using Target System Memory

Assuming that the user memory system has been tested, the next step is to load the full program into the users memory. If the program is to be in ROM or PROM then the PROM's should be programmed and installed in the user prototype. If the program is in RAM it can be loaded into the user system RAM from the disk.

The U (USERMAP) command (or in the case of the 8080 ICE the memory card straps) can be used to direct the required memory cycles to the user system rather than to the AMDS thus allowing access to user memory. For the 8080 it will be necessay to remove memory board jumpers in order to allow the system to direct memory accesses to the users memory.

The hardware breakpoint will be especially helpful during this phase because of the abilty to set a breakpoint without altering the contents of the program. Again it is very convenient if the user can incorporate the AMDS debugging vectors into his program, as this will facilitate use of the debugging tools provided.

## System Operation With Microprocessor Installed

The final step in the process is, of course, to remove the ICE plug from the prototype board and replace it with the microprocessor. If the previous steps have been done carefully, then this a trivial step and the system will run immediately, just as it did with ICE plugged in place of the Microprocessor

COMMAND

M[mode][mode] ...

mode may be any combination of the following:

C Enables the user system clock line input(s). This mode causes the ICE to switch from the AMDS internal clock to the target system clock.

D Allows the target system to generate DMA requests to cause suspension of microprocessor operation. This allows user DMA logic to gain control of the bus and generate transactions to target system memory. (Note: the user DMA logic cannot access ROM/PROM simulator memory). For the 8080 processor the DMA request is caused by a high level (>2.4 VDC) on the HOLD line and the DMA grant signal is indicated by a high level on the HLDA line.

U Allows the target system to contol the following input lines: RESET, INTR, and READY. The normal (inactive) conditions for these lines are:

RESET = low (<.4 VDC)
INTR = low
READY = high

Note especially: that a low on the READY line will completely halt the microprocessor.

The U mode parameter also enables the emulator to direct I/O requests to the target system. For the 8080 emulator the user must be aware of the I/O ports that the system uses. (see the section on 8080 ICE features).

E The E mode is a combination of C, U, and D.

R The R mode will disable the CRT and Memory refresh during user program execution. Of course it is not possible to use this mode parameter without the static RAM option, since disrupting the refresh will cause both the user program and the debugger to be lost if they are stored in dynamic RAM.

If the M command is entered without any mode parameters, then the system will return to full host mode and all inputs, clocks, Memory and I/O requests will be directed to the host system.

The user should be aware of the exact timing of the various mode switches. The C, U, and D, mode parameters will cause an immediate switch of the affected control lines as soon as the Carriage Return is typed following MC, MCU, MCD etc. The R mode parameter does not switch off the refresh function until the the user program is executed. Then the refresh function is switched back on as soon as control is returned to the debugger after a break from the user program.

COMMAND

M[mode][mode] ...

mode may be any combination of the following:

C Enables the user system clock line input(s). This mode causes the ICE to switch from the AMDS internal clock to the target system clock.

D Allows the target system to generate DMA requests to cause suspension of microprocessor operation. This allows user DMA logic to gain control of the bus and generate transactions to target system memory. (Note: the user DMA logic cannot access ROM/PROM simulator memory). For the 8080 processor the DMA request is caused by a high level (>2.4 VDC) on the HOLD line and the DMA grant signal is indicated by a high level on the HLDA line.

M Enables the memory mapping function. (see the USERMAP command).

I Allows the target system to contol the following input lines: RESET, INTR, and READY. The normal (inactive) conditions for these lines are:

```
RESET = low (<.4 VDC)
INTR  = low
READY = high
```

Note especially: that a low on the READY line will completely halt the microprocessor.

The I mode parameter also enables the emulator to direct I/O requests to the target system. For the 8080-1 emulator the user must be aware of the emulator control port (output X'F6').

E The E mode is a combination of C, I, M, and D.

R The R mode will disable the CRT and Memory refresh during user program execution. Of course it is not possible to use this mode parameter without the static RAM option, since disrupting the refresh will cause both the user program and the debugger to be lost if they are stored in dynamic RAM.

If the M command is entered without any mode parameters, then

the system will return to full host mode and all inputs, clocks, Memory and I/O requests will be directed to the host system.

The user should be aware of the exact timing of the various mode switches. The C, and D mode parameters will cause an immediate switch of the affected control lines as soon as the Carriage Return is typed following MC, MD, MCD etc. The R mode parameter does not switch off the refresh function until the the user program is executed. Then the refresh function is switched back on as soon as control is returned to the debugger after a break from the user program. The I mode parameter will enable the interrupt, reset, ready control lines and I/O requests when the user program is executed and disable them when the control is returned to the debugger. The M mode parameter will enable the memory mapping function whenever a memory access is made (for example to display memory, alter memory, execute a user program etc.).

COMMAND

M[mode][mode] ...

mode may be any combination of the following:

C Enables the user system clock line input(s). This mode causes the ICE to switch from the AMDS internal clock to the target system clock.

D Allows the target system to generate DMA requests to cause suspension of microprocessor operation. This allows user DMA logic to gain control of the bus and generate transactions to target system memory. (Note: the user DMA logic cannot access ROM/PROM simulator memory). For the 8085 processor the DMA request is caused by a high level (>2.4 VDC) on the HOLD line and the DMA grant signal is indicated by a high level on the HLDA line.

M Enables the memory mapping function. (see the USERMAP command).

I Allows the target system to contol the following input lines: RESET-, INTR, TRAP, RST 6.5, RST 7.5 and READY. The normal (inactive) conditions for these lines are:

```
RESET- = high (>2.4 VDC)
INTR   = low  (<0.4 VDC)
READY  = high
TRAP   = low
RST6.5 = low
RST7.5 = low
```

Note especially: that a low on the READY line will completely halt the microprocessor.

The I mode parameter also enables the emulator to direct I/O requests to the target system. For the 8085 emulator the user must be aware of the emulator control port (output X'F6').

E The E mode is a combination of C, I, M, and D.

R The R mode will disable the CRT and Memory refresh during user program execution. Of course it is not possible to use this mode parameter without the static RAM option, since disrupting the refresh will cause both

the user program and the debugger to be lost if they are stored in dynamic RAM.

If the M command is entered without any mode parameters, then the system will return to full host mode and all inputs, clocks, Memory and I/O requests will be directed to the host system.

The user should be aware of the exact timing of the various mode switches. The C, and D mode parameters will cause an immediate switch of the affected control lines as soon as the Carriage Return is typed following MC, MD, MCD etc. The R mode parameter does not switch off the refresh function until the the user program is executed. Then the refresh function is switched back on as soon as control is returned to the debugger after a break from the user program. The I mode parameter will enable the interrupt, reset, ready control lines and I/O requests when the user program is executed and disable them when the control is returned to the debugger. The M mode parameter will enable the memory mapping function whenever a memory access is made (for example to display memory, alter memory, execute a user program etc.).

COMMAND

M[mode][mode] ...

mode may be any combination of the following:

C Enables the user system clock line input(s). This mode causes the ICE to switch from the AMDS internal clock to the target system clock.

D Allows the target system to generate DMA requests to cause suspension of microprocessor operation. This allows user DMA logic to gain control of the bus and generate transactions to target system memory. (Note: the user DMA logic cannot access ROM/PROM simulator memory). For the Z80 processor the DMA request is caused by a low level (<0.4 VDC) on the BUSRQ- line and the DMA grant signal is indicated by a low level on the BUSAK- line.

M Enables the memory mapping function. (see the USERMAP command).

I Allows the target system to contol the following input lines: RESET-, INT-, WAIT-, and NMI-. The normal (inactive) conditions for these lines are:

```
RESET- = high (>2.4 VDC)
INT-   = high
WAIT-  = high
NMI-   = high
```

Note especially: that a low on the WAIT- line will completely halt the microprocessor.

The I mode parameter also enables the emulator to direct I/O requests to the target system. For the Z80 emulator the user must be aware of the emulator control port (output X'F6').

E The E mode is a combination of C, I, M, and D.

R The R mode will disable the CRT and Memory refresh during user program execution. Of course it is not possible to use this mode parameter without the static RAM option, since disrupting the refresh will cause both the user program and the debugger to be lost if they are stored in dynamic RAM.

If the M command is entered without any mode parameters, then the system will return to full host mode and all inputs, clocks, Memory and I/O requests will be directed to the host system.

The user should be aware of the exact timing of the various mode switches. The C, and D mode parameters will cause an immediate switch of the affected control lines as soon as the Carriage Return is typed following MC, MD, MCD etc. The R mode parameter does not switch off the refresh function until the the user program is executed. Then the refresh function is switched back on as soon as control is returned to the debugger after a break from the user program. The I mode parameter will enable the interrupt, reset, ready control lines and I/O requests when the user program is executed and disable them when the control is returned to the debugger. The M mode parameter will enable the memory mapping function whenever a memory access is made (for example to display memory, alter memory, execute a user program etc.).

COMMAND

M[mode][mode] ...

mode may be any combination of the following:

C Enables the user system clock line input(s). This mode causes the ICE to switch from the AMDS internal clock to the target system clock.

D Allows the target system to generate DMA requests to cause suspension of microprocessor operation. This allows user DMA logic to gain control of the bus and generate transactions to target system memory. (Note: the user DMA logic cannot access ROM/PROM simulator memory). For the 6800 processor the DMA request is caused by a low level (<0.4 VDC) on the HALT line and the DMA grant signal is indicated by á high level (>2.4 VDC) on the BA line. The D mode parameter also enables the TSC and DBE control lines.

U Enables the memory mapping function. (see the USERMAP command).

I Allows the target system to contol the following input lines: IRQ-, NMI-, and RESET-. The normal (inactive) conditions for these lines are:

IRQ- = high
NMI- = high
RESET- = high

E The E mode is a combination of C, I, U, and D.

R The R mode will disable the CRT and Memory refresh during user program execution. Of course it is not possible to use this mode parameter without the static RAM option, since disrupting the refresh will cause both the user program and the debugger to be lost if they are stored in dynamic RAM.

P The P mode will not allow the user program to access system I/O.

If the M command is entered without any mode parameters, then the system will return to full host mode and all inputs, clocks, Memory and I/O requests will be directed to the host system.

The user should be aware of the exact timing of the various

mode switches. The C, and D mode parameters will cause an immediate switch of the affected control lines as soon as the Carriage Return is typed following MC, MD, MCD etc. The R mode parameter does not switch off the refresh function until the the user program is executed. Then the refresh function is switched back on as soon as control is returned to the debugger after a break from the user program. The I mode parameter will enable the interrupt, and reset control lines only when the user program is executed and disable them when the control is returned to the debugger. The U mode parameter will enable the memory mapping function whenever a memory access is made (for example to display memory, alter memory, execute a user program etc.).

COMMAND

M[mode][mode] ...

mode may be any combination of the following:

C Enables the user system clock line input(s). This mode causes the ICE to switch from the AMDS internal clock to the target system clock.

D Allows the target system to generate HALT requests to cause suspension of microprocessor operation. This allows user DMA logic to gain control of the bus and generate transactions to target system memory. The 6802 does not tri-state its buses, so the user must implement the tri-state buses in the target system. (Note: the user DMA logic cannot access ROM/PROM simulator memory). For the 6802 processor the HALT request is caused by a low level (<0.4 VDC) on the HALT- line and the DMA grant signal is indicated by a high level (>2.4 VDC) on the BA line.

U Enables the memory mapping function. (see the USERMAP command).

I Allows the target system to contol the following input lines: IRQ-, NMI-, MR and RESET-. The normal (inactive) conditions for these lines are:

```
IRQ-   = high
NMI-   = high
MR     = high
RESET- = high
```

E The E mode is a combination of C, I, U, and D.

R The R mode will disable the CRT and Memory refresh during user program execution. Of course it is not possible to use this mode parameter without the static RAM option, since disrupting the refresh will cause both the user program and the debugger to be lost if they are stored in dynamic RAM.

P The P mode will not allow the user program to access system I/O.

If the M command is entered without any mode parameters, then the system will return to full host mode and all inputs, clocks, Memory and I/O requests will be directed to the host system.

The user should be aware of the exact timing of the various mode switches. The C, and D mode parameters will cause an immediate switch of the affected control lines as soon as the Carriage Return is typed following MC, MD, MCD etc. The R mode parameter does not switch off the refresh function until the the user program is executed. Then the refresh function is switched back on as soon as control is returned to the debugger after a break from the user program. The I mode parameter will enable the interrupt, and reset control lines only when the user program is executed and disable them when the control is returned to the debugger. The U mode parameter will enable the memory mapping function whenever a memory access is made (for example to display memory, alter memory, execute a user program etc.).

COMMAND

M[addr],{mask}

Transfers the memory space specified by "mask" starting at "addr" into the user system. Whenever a memory location is accessed that has been mapped into the user system, the memory request and read or write will be directed to the user system instead of the host ROM/PROM simulator. All addresses may be mapped into the user system except those priviledged locations reserved by the debugger communication block and in some cases the emulator control ports. See the section "Emulator Features".

Each user map begins on a 2K boundary. Each bit in the mask will switch 256 bytes into the user system. The bits are individually "or'ed" to form the mask. Thus U1800,0F would switch the memory space from X'1800' to X'1BFF' into the user system. If "addr" is not on a 2K boundary then the message "NOT ON 2K BOUNDARY" will be displayed. If the "addr" parameter is omitted then the map will begin at the present cursor location.

2K Boundaries

| | | | |
|---|---|---|---|
| 0000 | 4000 | 8000 | C000 |
| 0800 | 4800 | 8800 | C800 |
| 1000 | 5000 | 9000 | D000 |
| 1800 | 5800 | 9800 | D800 |
| 2000 | 6000 | A000 | E000 |
| 2800 | 6800 | A800 | E800 |
| 3000 | 7000 | B000 | F000 |
| 3800 | 7800 | B800 | F800 |

COMMAND

N[expression]

Execution starts at the address specified by the value of the expression. If the expression is omitted, execution starts at the memory pointer address of the active side of the display. When execution starts, all maskable interrupts are disabled. Unless the user enables interrupts, the BREAK key, and hardware breakpoint will have no effect on the 8080 and 8085 processor. This form of the execute command is very useful when debugging interrupt systems and software.

N$

Execution starts at the address specified in the program counter (register display).

COMMAND Object Program Load

L{file-name}[,[offset][,symbol-table-address]]

Loads "file-name" into the address space specified in the object code records, modified by the optional "offset". If a symbol table is appended to the object file and the "symbol-table-address" parameter is given, then the symbol table will also be loaded.

COMMAND Display Control

TAB

The TAB command will split the screen into to separately addressable halves, and then select which side of the screen is to respond to the next command.

COMMAND . Display Memory

D{expression}

Display the contents of memory around the address obtained by evaluating "expression".

D$

Return the previous display to the screen.

COMMAND Cursor Control

↓ Advance the display 1 line.
↑ Backup the display 1 line.
-→ Advance the display 1 byte.
-← Backup the display 1 byte.

COMMAND Store Data

S{constant}[,constant] ...

The "constants" are stored, one byte at a time, starting at the display cursor address.

COMMAND Alter Registers

Z{reg=constant}[,reg=constant] . . .

Alter the contents of the microprocessor registers.

COMMAND Find Data String

F[constant][,constant] . . .

Search memory for the constant string specified. If no constant is specified, the last value for the string comparison will be reused.

COMMAND Breakpoint

BS{n}

Set breakpoint "n" at the present display cursor location.

BD{n}

Display the memory location corresponding "n".

BR

Clear the breakpoint (if any) at the cursor location.

BC

Clear the breakpoint data save area. Used when the the object program is re-loaded to remove all the debugger data about the previous breakpoints.

COMMAND Execute Program

E[expression]

Begin user program execution at the location computed from "expression".

E$

Begin user program execution at the location stored in the "PC" in the register display.

STEP

The STEP will execute a single instruction every time it is used. Note: a program that disables interrupts will render the STEP key non-functional. When the "R" parameter is used in conjunction with the "M" (Mode) command, the STEP key cannot be used.

COMMAND Write Object Data to File

W{file-name}{,st-addr,end-addr}[,st-addr,end-addr]...[ent-addr]

Write absolute object data to "file-name" starting at the first "st-addr" and "end-addr" and ending with the last "st-addr" and "end-addr". The last entry, "ent-addr" provides an entry address for the file.

COMMAND Trace Execution

T[expression]

Trace execution of the program starting at "expression". When the user program is halted, the last 20 processor cycles will be displayed, including all registers, stack pointer, flags and program counter. If "expression" is omitted, then execution will be at the display cursor location.

T$

Start trace execution at the location specified by the program counter in the register display.

TR

Recall the last trace display.

COMMAND Alter Protection

X

The "X" command switches write protection on or off. It will effect the 1024 (X'400') byte block pointed to by the display cursor.

C[type][,length]

Tests an EPROM for the empty condition. The parameter "type" may be 2704, 2708, 2758, 2716 or 2732. The proper personality module must be installed, of course. The first "length" locations will be tested.

P[type][,length]

Program an EPROM. The EPROM will be programmed with the data starting at the present display cursor location.

I[type][,length]

Input the contents of an EPROM starting at the present display cursor location.

V[type],[,length]

Data in the EPROM will be compared (verified) to the data in memory, starting at the present cursor location.

Installation of the 8080 ICE package involves the following steps:

(1) Two short jumper cables are included to connect the ICE interface card and the 8080 CPU card.

(2) Two I.C.'s must be removed from the 8080 CPU card (U13 -- 8T97 and U21 74S161 or 93S16).

(3) These two I.C.'s will be replaced by the short jumper cables, which will connect to the back side of the ICE interface card.

(4) After connecting the ICE interface card and 8080 CPU card, lay the two cards face up on a table with the edge fingers to the left. The ICE interface card will be on top. Lay the 8080 interface probe assembly, face down, with the probe connector to the right. Face down implies that the component side of the probe card is on the bottom and the circuit side is facing up. The topmost cable connects to J1 (the further connector) on the ICE interface card. The lower cable connects to J2 (the closer cable) on the ICE interface card.

(5) Now install the CPU card and ICE interface card in the AMDS. You will have to allow one empty slot between the two cards.

(6) The cables between the ICE interface card and the probe assembly can now be folded and brought out through the back of the unit using one of the cable clamps.

Installation of the 8080-1 ICE involves the following steps:

(1) Remove jumper connectors __, __, and __ on the 8080-1 CPU card. This will allow gating of memory requests and clock signals via the ICE interface card.

(2) The short __ pin cable is used connect J_ on the 8080-1 CPU card and J_ on the ICE interface card.

(3) After connecting the 8080-1 CPU card and the ICE interface card, lay the two cards on the table, face up with the edge finger connectors to the left. The ICE interface card will be on top. Lay the ICE probe assembly on the table with the probe connector to the right. Connect the topmost cable from the probe assembly to J_ (the lower or leftmost of the two connectors) on the ICE interface card. Connect the lowermost cable from the probe assembly to J_ (the upper or rightmost of the two connectors) on the ICE interface card.

(4) Now install the 8080-1 CPU card and ICE interface card in the AMDS mainframe in adjacent slots.

(5) The two cables that connect the ICE probe assembly and the ICE interface card can be folded and brought out the rear of the AMDS through on of the cable clamps at the bottom of the AMDS case.

Installation of the 8085 ICE involves the following steps:

(1) Remove jumper connectors C, D, and E on the 8085 CPU card. This will allow gating of memory requests and clock signals via the ICE interface card.

(2) The short 40 pin cable is used to connect J3 on the 8085 CPU card and J3 on the ICE interface card. The short 20 pin cable is used to connect J4 on the 8085 CPU card and J4 on the ICE interface card.

(3) After connecting the 8085 CPU card and the ICE interface card, lay the two cards on the table, face up with the edge finger connectors to the left. The ICE interface card will be on top. Lay the ICE probe assembly on the table with the probe connector to the right. Connect the topmost cable from the probe assembly to J1 (the lower or leftmost of the two connectors) on the ICE interface card. Connect the lowermost cable from the probe assembly to J2 (the upper or rightmost of the two connectors) on the ICE interface card.

(4) Now install the 8085 CPU card and ICE interface card in the AMDS mainframe in adjacent slots.

(5) The two cables that connect the ICE probe assembly and the ICE interface card can be folded and brought out the rear of the AMDS through on of the cable clamps at the bottom of the AMDS case.

Installation of the Z80 ICE involves the following steps:

(1) Remove jumper connectors D, G, and J on the Z80 CPU card. This will allow gating of memory requests and clock signals via the ICE interface card. If the system is equipped with high-speed static RAM then the jumper at A should be installed. This will delete a one-cycle wait state on all Z80 M1 cycles that are mapped into the ROM/PROM simulator memory. This wait state is necessary when using dynamic RAM at 4Mhz.

(2) The short 40 pin cable is used to connect J1 on the Z80 CPU card and J3 on the ICE interface card.

(3) After connecting the Z80 CPU card and the ICE interface card, lay the two cards on the table, face up with the edge finger connectors to the left. The ICE interface card will be on top. Lay the ICE probe assembly on the table with the probe connector to the right. Connect the topmost cable from the probe assembly to J1 (the lower or leftmost of the two connectors) on the ICE interface card. Connect the lowermost cable from the probe assembly to J2 (the upper or rightmost of the two connectors) on the ICE interface card.

(4) Now install the Z80 CPU card and ICE interface card in the AMDS mainframe in adjacent slots.

(5) The two cables that connect the ICE probe assembly and the ICE interface card can be folded and brought out the rear of the AMDS through on of the cable clamps at the bottom of the AMDS case.

Installation of the 6800 ICE involves the following steps:

(1) Cut the short traces at A, B, and C on the 6800 CPU card. This will allow gating of memory requests and clock signals via the ICE interface card.

(2) The short 20 pin cable is used to connect J1 on the 6800 CPU card and J3 on the ICE interface card.

(3) After connecting the 6800 CPU card and the ICE interface card, lay the two cards on the table, face up with the edge finger connectors to the left. The ICE interface card will be on top. Lay the ICE probe assembly on the table with the probe connector to the right. Connect the topmost cable from the probe assembly to J1 (the lower or leftmost of the two connectors) on the ICE interface card. Connect the lowermost cable from the probe assembly to J2 (the upper or rightmost of the two connectors) on the ICE interface card.

(4) Now install the 6800 CPU card and ICE interface card in the AMDS mainframe in adjacent slots.

(5) The two cables that connect the ICE probe assembly and the ICE interface card can be folded and brought out the rear of the AMDS through on of the cable clamps at the bottom of the AMDS case.

Installation of the 6802 ICE involves the following steps:

(1) Remove the jumper connectors at C, and M on the 6802 CPU card. This will allow gating of memory requests and clock signals via the ICE interface card.

(2) The short 40 pin cable is used to connect J3 on the 6800 CPU card and J3 on the ICE interface card.

(3) After connecting the 6802 CPU card and the ICE interface card, lay the two cards on the table, face up with the edge finger connectors to the left. The ICE interface card will be on top. Lay the ICE probe assembly on the table with the probe connector to the right. Connect the topmost cable from the probe assembly to J1 (the lower or leftmost of the two connectors) on the ICE interface card. Connect the lowermost cable from the probe assembly to J2 (the upper or rightmost of the two connectors) on the ICE interface card.

(4) Now install the 6802 CPU card and ICE interface card in the AMDS mainframe in adjacent slots.

(5) The two cables that connect the ICE probe assembly and the ICE interface card can be folded and brought out the rear of the AMDS through on of the cable clamps at the bottom of the AMDS case.

GenRad
Development Systems Division

**Technical Publications Remarks Form**

Please use this form to submit your suggestions for revisions, corrections, or additions to this publication. Your comments will be promptly investigated by appropriate technical personnel, and action will be taken as required. If your answer to any of the questions is 'NO' or requires qualification, please include any additional comments on a separate sheet and enclose it inside this pre-addressed form.

TITLE: _____

ORDER NO. _____

DATED _____

| | YES | NO |
|---|---|---|
| DOES THIS DOCUMENT MEET YOUR NEEDS? | _____ | _____ |
| IS THE MATERIAL CONTAINED IN THIS DOCUMENT: | | |
| ACCURATE? | _____ | _____ |
| EASY TO READ AND UNDERSTAND? | _____ | _____ |
| ORGANIZED FOR CONVENIENT REFERENCING? | _____ | _____ |
| WELL ILLUSTRATED WITH USEFUL EXAMPLES? | _____ | _____ |
| COMPLETE? | _____ | _____ |

ERRORS IN THE PUBLICATION

SUGGESTIONS FOR IMPROVEMENT

(Please Print)

FROM: NAME _____ DATE _____

ORGANIZATION _____

ADDRESS _____

TITLE _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 608 CULVER CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

GenRad

Development Systems Division

5730 Buckingham Parkway
Culver City, CA 90230

ATTN: MARKETING TECHNICAL PUBLICATIONS